PlayStation Suite

A new environment for open development

Ron Schaffner - Sony Computer Entertainment America
Chris Norden - Sony Computer Entertainment America

Agenda

- Business Overview
  - Concept
  - SDK Licensing Model
  - Content Submission and Approval Process
  - PlayStation Suite / PlayStation Store
  - Target Devices
  - Developer Community
  - Roadmap
- Technical Overview

Concept

- “Application Store” Model
  - An open environment
    - New approach for SCE, targeting all ranges of developers (traditional developers to independent developers)
  - Casual game focused (non-game applications also considered)
    - Lightweight content approval process
  - Distinct from existing PSN content but available on all PS Certified devices and PlayStation Vita
- PlayStation Suite SDK
  - Cross platform / cross device, binary level compatibility
  - Helping to address device fragmentation issues

SDK Licensing

- Accessible/Downloadable by virtually anybody
  - Minimal screening required
- Target Developers – Homebrew all the way up to professional
- PS Suite SDK
  - C# as primary programming language
  - Program will sit on top of a virtual machine which is supported by PS Certified Android devices and PS Vita
  - Includes 3D graphics libraries for games and UI Toolkit for non-game application development

Content Submission and Approval Process

- Objectionable content – Guidelines and takedown procedures
- Review / Approval – Exploring the balance between quick iteration times and a curated experience
- Publishing Process – Self service, single submission

Business Model

- One global submission
- Choose from pre-selected wholesale price tiers
  - In-game ads are NOT allowed
  - Links to outside sales are NOT allowed
- US Example:
  - If a developer selects a wholesale price tier of USD$6.99, the PS Store retail price* will be set around USD$9.99
  - *The retail price will be set at the retailer's discretion

PS Store for PS Suite

- Separate storefront that spans platforms
- Focus on content discovery and promotion
- In-game purchases will be allowed (through the PSN Wallet)
- Aggressively support business models that work
- Return policy is under discussion
- Minimum QA will be done by SCE

PS Store for PS Suite

- The PS Store is already available in the following nine countries, and phased rollout of the update will start later this year
    - United States
    - Canada
    - United Kingdom
    - France
    - Italy
    - Germany
    - Spain
    - Japan
    - Australia
    - More countries to be added at a later date

Target Devices

- PS Vita
  - Japan/Asia launch – December 17, 2011
  - US/Europe launch – February 22, 2012
- PS Certified Android devices
  - Sony Xperia series
  - Sony Tablet
  - More to come (Non-Sony devices included)

Developer Community

- Community-driven, forum-based support
- Sharing-friendly
  - Create demos, samples, or libraries, and freely share them with anybody
  - Talk about PS Suite development openly and freely with anybody on forums, blogs, etc.

Roadmap

- From April 2012, Developer Portal (SDK and forums) will be available to everyone
- Available countries will be updated at a later date (via newsletter)
- Present participants of the closed beta test will receive an update when ready through the registered email address

Roadmap

- SDK 0.98 will be released in April 2012
  - No license fee
  - Fully testable on PS Vita
- SDK (official release) will be released later this year
  - $99 license fee (per year)
  - Other territories will have a similar license fee
  - Submissions will be enabled
- Other PSN features are under consideration for future releases

Technical Overview

Technical Overview

- Cross-platform game development environment
  - PS Vita, Android, etc…
  - Binary level compatibility across all platforms – runs the same .exe on every device!
- C#
  - Modern, high-level programming language
- Development on simulator or actual device
  - There's no need to purchase or prepare a special development tool
  - This is a big change from existing PlayStation development requirements
- Better support for games
  - Contains all the necessary libraries and tools for efficient game development
- Also supports the creation of non-game applications
  - Contains UI library and UI design tools to make complex GUI development less painful

Programming language

- Main programming language is C#
  - Modern, high-level programming language
  - Runs on a Mono virtual machine which provides direct binary compatibility across devices
  - Compile a single .exe file, then run it on any of the supported devices without modification
- Using native code is not allowed
  - Provides better security, more predictable output, and cross-platform portability
  - Performance is excellent for a wide range of games and applications
- SCE may support other languages in the future
  - Based on user feedback and if it makes sense…

Demo

- Games
- Non-game applications
- Devices shown (remember, the same .exe is running on every device!)
  - PC Simulator
  - PS Vita
  - Sony Ericsson Xperia PLAY
  - Sony Xperia S (NX)
  - Sony Tablet S
- Tools
  - PS Suite Studio (IDE)
  - PS Suite UI Composer

SDK Contents

- PS Suite Studio (IDE based on MonoDevelop)
  - Debugger
  - Project Template
- PS Suite UI Composer (UI Design Tool)
- Simulator
- Library (Core APIs, High Level APIs)
- C# Toolchain (compiler, linker, etc.)
- Documentation
  - Development guide
  - API Reference
- Samples
- Demo Games, Demo Applications

- Note: Currently, only Windows environments are supported

Screenshot – Studio (IDE)

RpgDemo - src¥scene¥SceneDataLoad.cs - PlayStation Suite Studio

File Edit View Search Project Build Run Version Control Tools Window Help

PlayStation Suite Simulator | Release

Solution

- Solution RpgDemo
  - BasicModel
  - DemoGameLib
  - RpgDemo
    - References
    - res
    - shaders
    - src
      - actor
      - cmn
      - ctrl
      - data
      - obj
      - scene
        - RpgSetupData
        - TutorialData
        - SceneDataLoad.cs
        - SceneDebugMenu.cs
        - SceneDebugObjPlac
        - SceneGameMain.cs
        - SceneGameResult.c
        - SceneTitle.cs
        - SceneTutorial.cs
      - shape
      - AppDebug.cs
      - AppDispEff.cs
      - AppInput.cs
      - AppLyout.cs
      - AppSound.cs
      - RpgDemo.cs
    - app.cfg

Solution | Classes

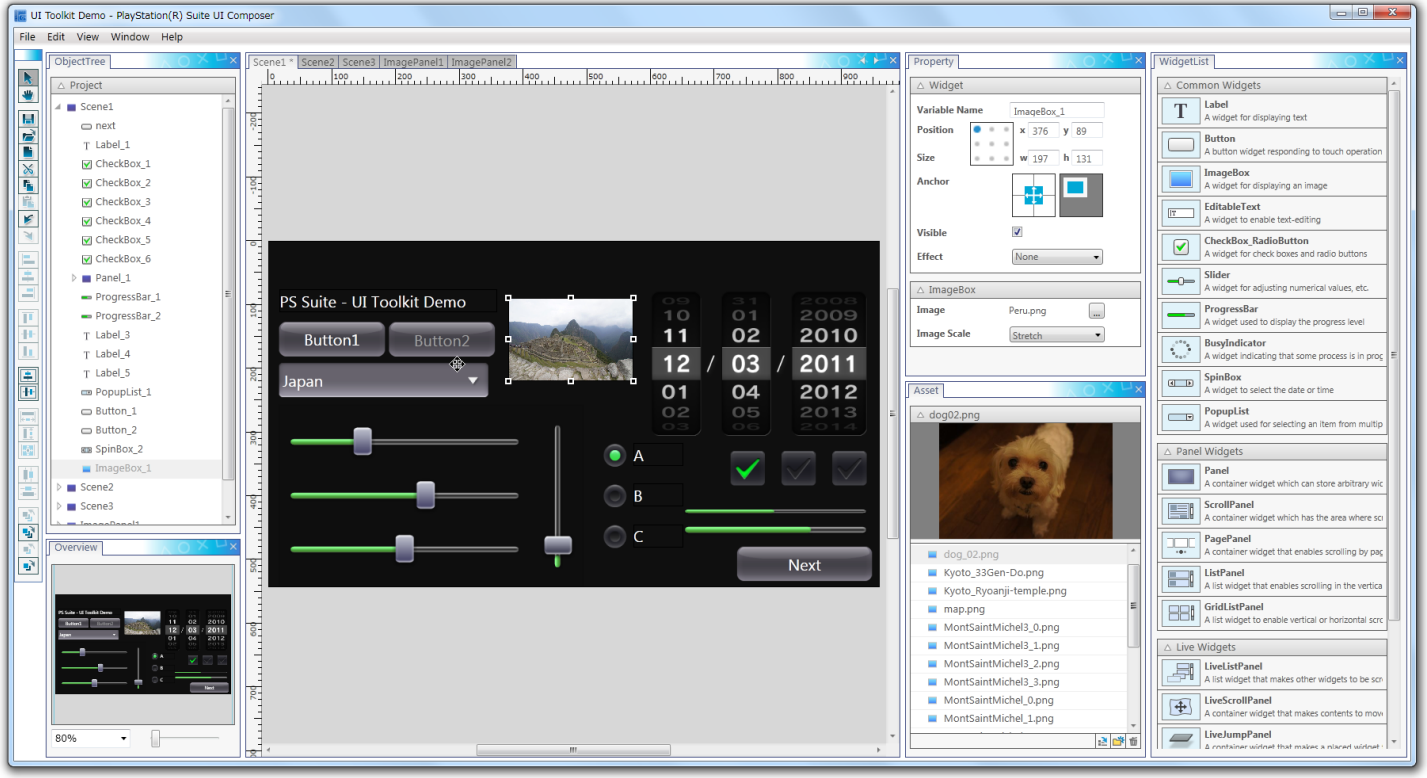
RpgDemo.cs × | AppInput.cs × | SceneDataLoad.cs ×

No selection

```
        useGraphDev.Graphics.SetClearColor( 0.0f, 0.020f, 0.20f, 0.0f ) ;
        useGraphDev.Graphics.Clear() ;

        DemoGame.Graphics2D.RemoveSprite( "mess" );
        DemoGame.Graphics2D.AddSprite( "mess", " Now Loading ...", 0xffffffff, 20, 120 );
        DemoGame.Graphics2D.DrawSprites();

        AppDispEff.GetInstance().Draw( useGraphDev );

        useGraphDev.Graphics.SwapBuffers();

        DemoGame.Graphics2D.RemoveSprite( "mess" );
        return true;
    }



/// ...

    private bool loadModelData()
    {
        SetupModelData    setupData = new SetupModelData();

        if( setupData.SetStgData() == false ){
            return false;
        }
        if( setupData.SetCharData() == false ){
            return false;
        }
        if( setupData.SetEquipData() == false ){
            return false;
        }
        if( setupData.SetEffData() == false ){
            return false;
        }
        if( setupData.SetFixData() == false ){
            return false;
        }

        setupData = null;
        return true;
    }


    private bool loadParamData()
    {
        SetupChMvtData    setupData = new SetupChMvtData();

        if( setupData.SetHeroData() == false ){
            return false;
        }
```

Ready | 1 : 1 | INS | Feedback | 0 0 | Task List

Screenshot – UI Composer (UI Design Tool)



Screenshot – PC Simulator

Screenshot – Documentation

Firefox

PS Suite SDK: Graphics Overview

PS Suite SDK

Search

- PS Suite SDK
  - PS Suite SDK Document Index
  - Related Pages
    - Development Guide
    - API Overview
      - Core Overview
      - Graphics Overview
      - Audio Overview
      - Input Overview
      - Imaging Overview
      - Environment Overview
      - HighLevel.UI Overview
      - HighLevel.Model Overview
    - Other Informations
  - Class List
  - Class Index
  - Class Hierarchy
  - Class Members
  - Packages
  - Package Functions

Graphics Overview

Sce.Pss.Core.Graphics provides a feature for graphics rendering.

Its functional model is based on OpenGL ES 2.0.

GraphicsContext

GraphicsContext controls overall graphics rendering; this class represents the graphics context of OpenGL. The constructor of this class is used to initialize the graphics device and the screen; the member functions of this class are used to carry out graphics rendering.

GraphicsContext initializes the graphics device and the screen according to the system's standard settings. If necessary, the screen size and pixel format can be specified as arguments. Specifications of the graphics device and actual screen size, as well as the pixel format, can be obtained after initialization.

To carry out graphics rendering, create a graphics object (described later) in advance, set the graphics object and render states, and render the primitive using the GraphicsContext member function.

ShaderProgram

ShaderProgram controls shader features; this class represents program objects of OpenGL. Consisting of the vertex shader and the fragment shader, this class is created from the file storing these shader programs.

ShaderProgram holds multiple uniform variables and attribute variables. Values can be set to the uniform variables, and vertex data can be continuously transferred to the attribute variables. These variables are accessed by the numbers assigned to them. If necessary, these variables can be linked to specific numbers in a manner that is useful to the application.

The extension of the shader file is *.cgx, and it is binary converted from source code written in Cg language. Use *.vcg and *.fcg or *.vp.cg and *.fp.cg for the extensions of the source code. When the extensions of the source code are *.vcg and *.fcg, a vertex shader and fragment shader with the same name are stored in one file.

VertexBuffer

VertexBuffer stores vertex data; this class represents vertex buffer objects of OpenGL. This class can store multiple vertex arrays, 1 index array, and 1 primitive array, as necessary; it can be created by specifying the size and format.

Multiple VertexBuffer can be specified to a GraphicsContext. Each vertex array is linked to an attribute variable of the ShaderProgram in order.

When multiple index arrays or primitive arrays are enabled, the last one will be used. A primitive array can be specified as an argument of the rendering API instead of being stored in the VertexBuffer; however, rendering may be optimized if the primitive array is stored in the VertexBuffer.

Texture

Texture stores pixel data; this class represents an texture object of OpenGL. Derivative classes (Texture2D, TextureCube, etc.) exist; a Texture can be created from an image file or by specifying its size and format.

Screenshot – Developer Forum

Firefox

PlayStation Suite Developer Community -...

Developer Program for PlayStation®Suite (Closed Beta Test) Forum

Follow

PlayStation Suite Developer Community

Forum : PlayStation Suite Developer Community

| Sign Out | My Settings

Category Search

Community

PlayStation Suite Developer Community

General (3 Items)

Title	Posts	New
Announcements & Events Latest Post – Re: GDC Demo opportunities	30	27
Suggestions Latest Post – UI suggestion regarding transitions	128	103
Community Lounge Latest Post – Re: Project Magnate progress update	148	137

APIs / Libraries (7 Items)

Title	Posts	New
General General Discussions about PS Suite SDK Latest Post – Re: Deployment on Vita	286	198
Core Some utility functions for Math, Vector and Color Latest Post – Re: Making full use of the 4 Cores...	9	6

Announcements

Welcome to the PlayStation Suite Developers Community

This is a community for the discussion of technical topics with other closed beta developers and SCE engineers. Posting ideas/requests are also appreciated. Join the discussion!

Top Kudoed Posts

Re: Project Magnate progress update	1
Re: Feedback so far...	1

View All

Virtual machine

- Runs on a Mono™ virtual machine
- A custom implementation of the Common Language Infrastructure (ECMA-335)
  - http://www.ecma-international.org/publications/standards/Ecma-335.htm
- Base Class Library (BCL) is API compatible with the core of the Portable Library project with some extensions to provide a better developer experience
  - http://msdn.microsoft.com/en-us/library/gg597391.aspx
- Supported items
  - Basic types like Array, String, Collections
  - File I/O
  - Threading
  - Sockets
  - Http
  - Xml
  - Etc...

PS Suite API

- Style is very straightforward
  - Class-structured
  - Simple
  - Easy to understand

```
// teapot
vbTeapot = new VertexBuffer( meshTeapot.VertexCount,
                             meshTeapot.IndexCount,
                             VertexFormat.Float3,
                             VertexFormat.Float2 );
vbTeapot.SetVertices( 0, meshTeapot.Positions );
vbTeapot.SetVertices( 1, meshTeapot.TexCoords );
vbTeapot.SetIndices( meshTeapot.Indices );


// shadow map(simple)
shaderShadowMap = new ShaderProgram( "shaders/Simple.cgx" );
shaderShadowMap.SetAttributeBinding( 0, "a_Position" );

// texture
shaderTexture = new ShaderProgram( "shaders/Texture.cgx" );
shaderTexture.SetAttributeBinding( 0, "a_Position" );
shaderTexture.SetAttributeBinding( 1, "a_TexCoord" );
texColorMap = new Texture2D( "data/renga.png", false );
texColorMap.SetWrap( TextureWrapMode.Repeat );
```

Core APIs

- Graphics
  - OpenGLES 2.0 equivalent
- Audio
  - SoundEffect
  - Bgm
- Input
  - Game Pad
  - Touch
  - Motion
- Imaging
  - Image Processing
  - Font
- VectorMath
  - Vector/Matrix calculation
- Environment
  - Clipboard
  - CommonDialog (TextInput, etc…)
  - Shell
  - SystemEvent
  - SystemParameters
  - Storage
  - PersistentMemory

Data formats

- Model
  - Custom format, but converter is provided for COLLADA, FBX, XSI, X
- Texture
  - PNG, JPEG, GIF, BMP (currently no DXT or PVRTC)
- Audio (SFX)
  - WAV (currently PCM only)
- Audio (BGM)
  - MP3

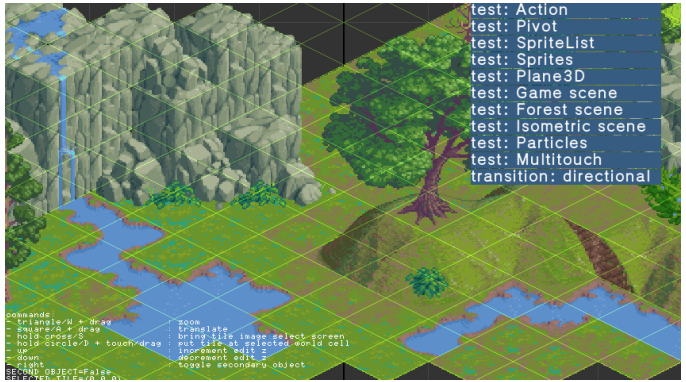
Shader language

- Developers can write their own custom shaders (vertex and fragment)
- Enables modern graphics programming across all platforms
- Basic shader samples are contained in the SDK

GameEngine2D

- Basic functionality needed to create simple 2D games
  - Sprite system, scene graph, scheduler, actions
  - API is conceptually similar to Cocos2D (http://www.cocos2d-iphone.org/)

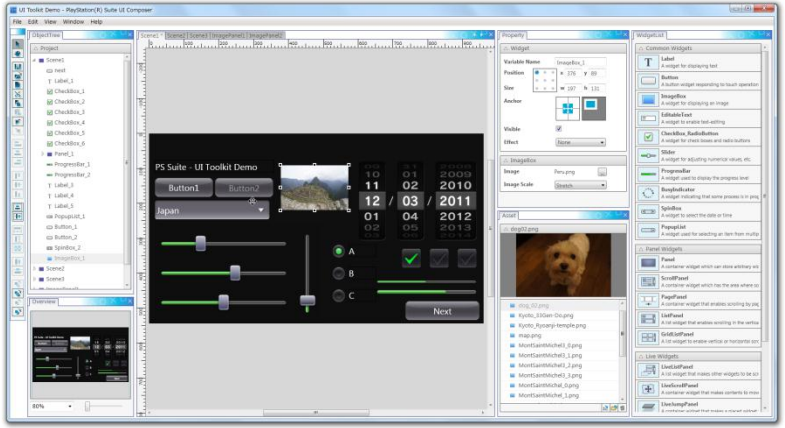
- Source code is provided



UI Toolkit / UI Composer

- Widget library and UI layout tool
  - Easy to use
  - Advanced look and feel (UI styling is based on PS Vita)

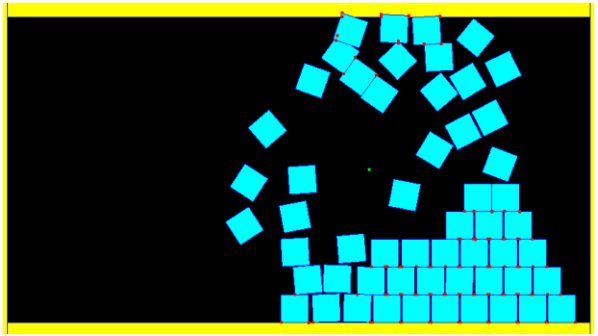
- Source code is provided



High-level APIs

- Model
  - Model rendering/animation

- Physics
  - 2D-based physics library



Graphics

Initialize

```
var vbuffer = new VertexBuffer(3, VertexFormat.Float3, VertexFormat.Float2 );
var program = new ShaderProgram("shader.cgx");
var texture = new Texture2D("texture.png", false);

float[] positions = {
    0.0f, 1.0f, 0.0f,
    -0.5f, 0.0f, 0.0f,
    0.5f, 0.0f, 0.0f,
};

vbuffer.SetVertices(0, positions);
```

Render

```
program.SetUniformValue(0, ref worldViewProj);

graphics.SetShaderProgram(program);
graphics.SetVertexBuffer(0, vbuffer);
graphics.DrawArrays(DrawMode.TriangleStrip, 0, 3);
```

Audio

Initialize

```
sound = new Sound("sound.wav");
soundPlayer = sound.CreatePlayer();
```

Play sound

```
soundPlayer.Volume = 0.8;
soundPlayer.Play();
```

Input.GamePad

GamePad

```
var gamePadData = GamePad.GetData(0);

var pressed = (gamePadData.Buttons & GamePadButtons.Circle) != 0;
var speed = new Vector2( gamePadData.AnalogLeftX, gamePadData.AnalogLeftY );
```

Input.Touch

Touch

```
var touchDataList = Touch.GetData(0);

for( var touchData in touchDataList ) {
    if (touchData.Status == TouchStatus.Down) {
        var position = new Vector2( touchData.X, touchData.Y );
    }
}
```

Input.Motion

Motion

```
var motionData = Motion.GetData(0);

Vector3 acceleration = motionData.Acceleration;
Vector3 anglularVelocity = motionData.AngularVelocity;
```

Imaging.Image

Decode

```
var image1 = new Image("image.png");
image1.Decode();
```

Modify

```
var image2 = image1.Resize( new ImageSize( 200, 150 ) );
```

Imaging.Font

Create

```
var font = new Font( FontAlias.System, 40, FontStyle.Regular );
```

Get metrics information

```
int width = font.GetTextWidth( text, 0, text.Length );
int height = font.Metrics.Height;
```

Render to Image

```
image.DrawText( text, color, position );
```

GameEngine2D

Construct Scene

```
var scene = new Scene();

var textureInfo = new TextureInfo( new Texture2D("texture.png", false ) );

var sprite = new SpriteUV();
sprite.TextureInfo = textureInfo;
scene.AddChild( sprite );
```

Run main loop

```
Director.Instance.RunWithScene( scene );
```

Action

```
sprite.RunAction( new MoveTo( new Vector2(0.0f,0.0f), 0.1f ) );
sprite.RunAction( new ScaleBy( new Vector2(2.0f,2.0f), 0.1f ) );
```

Model

Load

```
model = new BasicModel( "walker.mdx", 0 );
program = new BasicProgram() ;
```

Animation and Rendering

```
model.SetWorldMatrix( ref world ) ;
model.Animate( stepTime ) ;
model.Update() ;
model.Draw( graphics, program ) ;
```

UI

Initialize and construct Scene

```
UISystem.Initialize(graphics);

Scene scene = new Scene();

Label label = new Label();
label.Text = "Hello World!";

scene.RootWidget.AddChildLast(label);
```

UI

Update

```
UISystem.Update(touchDataList);
```

Render

```
UISystem.Render();
```

Event Handler

```
button1.ButtonAction += (sender, e) => {
    Dialog d = new Dialog();
    d.Show(new SlideInEffect());
};
```

Technology Roadmap

- Features coming in the future (date to be announced)
- Visual Studio integration

- Features that are under discussion (not yet planned)
- Support for additional programming languages
- Low-level audio API
- Camera API
- Location API

- Other features to be driven by community feedback